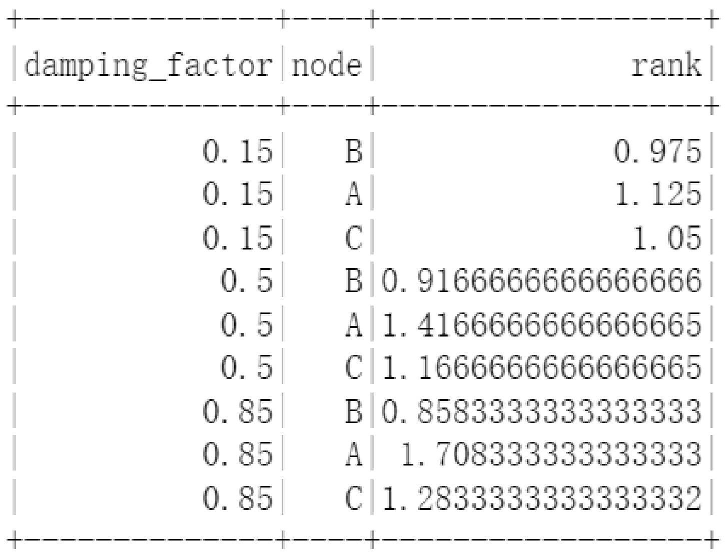
Contents

1 Results

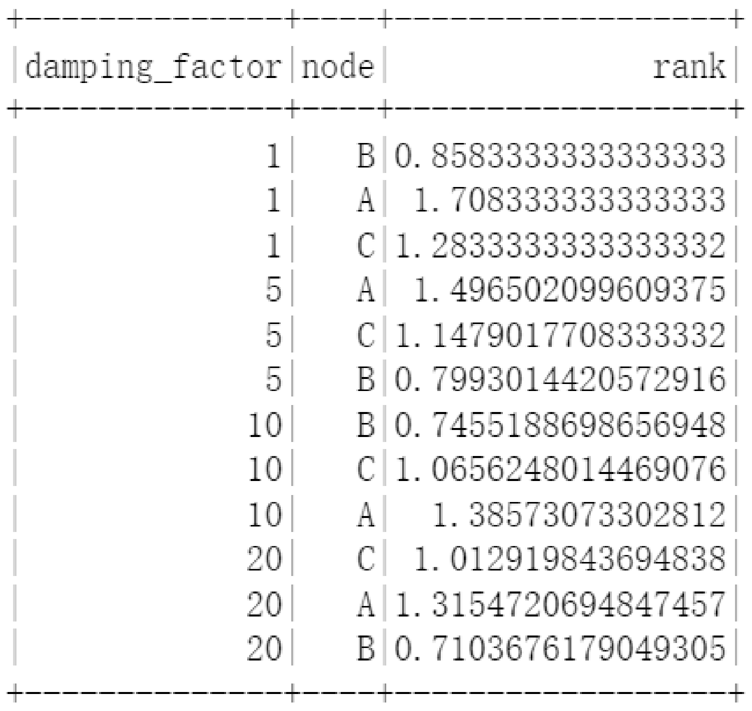
1.1 Table 1: Different Contribution Values



```
+--------------+----+------------------+
|damping_factor|node|              rank|
+--------------+----+------------------+
|          0.15|   B|             0.975|
|          0.15|   A|             1.125|
|          0.15|   C|              1.05|
|           0.5|   B|0.9166666666666666|
|           0.5|   A|1.4166666666666665|
|           0.5|   C|1.1666666666666665|
|          0.85|   B|0.8583333333333333|
|          0.85|   A| 1.708333333333333|
|          0.85|   C|1.2833333333333332|
+--------------+----+------------------+
```

Figure 1: Table1 SnapShot

1.2 Table 2: Different Iterations



```
+--------------+----+------------------+
|damping_factor|node|              rank|
+--------------+----+------------------+
|             1|   B|0.8583333333333333|
|             1|   A| 1.708333333333333|
|             1|   C|1.2833333333333332|
|             5|   A| 1.496502099609375|
|             5|   C|1.1479017708333332|
|             5|   B|0.7993014420572916|
|            10|   B|0.7455188698656948|
|            10|   C|1.0656248014469076|
|            10|   A|  1.38573073302812|
|            20|   C| 1.012919843694838|
|            20|   A|1.3154720694847457|
|            20|   B|0.7103676179049305|
+--------------+----+------------------+
```

Figure 2: Table2 SnapShot

2 Why the Absence of Node 'D' in PageRank Results

In the context of the PageRank algorithm, as implemented in the provided Python script, the absence of node 'D' in the final results warrants a detailed explanation. This outcome is intrinsically linked to the foundational principles of the PageRank algorithm and the specific nature of the dataset being used.

1. **Initial Rank Assignment and Its Implications**: The initial setup of the algorithm assigns a uniform rank of 1.0 to each node. This is a standard approach to ensure an equal starting point for all nodes. However, for nodes that do not have any inbound links, such as node 'D' in our dataset, this initial rank remains unchanged throughout the iterative process. This unchanging state is due to the lack of rank contributions from other nodes, as node 'D' does not receive any inbound links in the given data structure.

2. **Core Mechanism of PageRank Contributions**: The essence of the PageRank algorithm is to redistribute ranks based on the network of inbound links. A node's rank increases when it receives a portion of the rank from nodes that link to it. In the dataset provided:

   ```
   A:B,C
   B:A,C
   C:A
   D:A,B,C
   ```

   We observe that while node 'D' links to other nodes ('A', 'B', and 'C'), it does not have any nodes linking to it. Consequently, throughout the iterations of the algorithm, 'D' does not accrue any additional rank from other nodes.

3. **Iteration Process and Lack of Rank Update for 'D'**: Each iteration of the algorithm recalculates the ranks based on the incoming links and their respective contributions. Since node 'D' is devoid of any inbound links, there is no contribution to its rank, and as a result, its rank remains constant at the initially assigned value.

4. **Result Collection and Interpretation**: The algorithm concludes with the collection of ranks for each node present in the ranks RDD. Node 'D', having no variation in its rank and maintaining its initial value, might appear to be excluded or neglected in the PageRank computation. However, it is crucial to understand that this is a direct consequence of the algorithm's mechanics and the specific structure of the input data, rather than an omission or error in the algorithm.

In essence, the absence of node 'D' from the updated ranks in the PageRank results is not an indication of its exclusion but rather a reflection of its position in the network topology — a node without inbound links. To address such scenarios, a more comprehensive implementation of the PageRank algorithm could incorporate a 'teleportation' factor or alternative mechanisms to account for nodes that are not referenced by others.

3 PySpark Code

```
def compute_pagerank(damping_factor, iterations, lines):
    # Parse the input lines into (node, [neighbors]) pairs
    links = lines.map(lambda urls: urls.split(":"))\
                 .map(lambda url_neighbors: (url_neighbors[0],
    url_neighbors[1].split(",")))\
                 .cache()

    # Initialize each node with a rank of 1.0
    ranks = links.map(lambda url_neighbors: (url_neighbors[0],
    1.0))

    def computeContribs(urls, rank):
        num_urls = len(urls)
        for url in urls:
            yield (url, rank / num_urls)

    for iteration in range(iterations):
        # Calculates URL contributions to the rank of other URLs.
        contribs = links.join(ranks).flatMap(
            lambda url_urls_rank:
    computeContribs(url_urls_rank[1][0], url_urls_rank[1][1]))

        # Re-calculates URL ranks based on neighbor
    contributions.
        ranks = contribs.reduceByKey(lambda x, y: x +
    y).mapValues(lambda rank: rank * damping_factor + (1 -
    damping_factor))

    return ranks.collect()

# Read data from S3
lines = sc.textFile("s3://bigdata-hw7-lanston/page-rank.txt")

# Experiment with different damping factors and iterations
results_damping = {damping_factor:
    compute_pagerank(damping_factor, 1, lines) for
    damping_factor in [0.15, 0.50, 0.85]}
results_iterations = {iteration: compute_pagerank(0.85,
    iteration, lines) for iteration in [1, 5, 10, 20]}


# Convert results to DataFrames for better display
from pyspark.sql import Row
def results_to_dataframe(results):
    rows = [Row(damping_factor=df, node=k, rank=v) for df, ranks
    in results.items() for k, v in ranks]
    return spark.createDataFrame(rows)


# DataFrame for different damping factors
df_damping = results_to_dataframe(results_damping)
df_damping.show()

# DataFrame for different iterations
df_iterations = results_to_dataframe(results_iterations)
df_iterations.show()
```

4 Summary of Spark-based PageRank Analysis

The study conducted using Apache Spark explores the dynamics of the PageRank algorithm, focusing on how various damping factors and iteration counts influence the ranking of web pages within a network. The analysis uses damping factors of 15%, 50%, and 85% to determine the impact on PageRank calculations. The findings demonstrate significant variations based on these parameters.

Influence of Damping Factor:

- **Low Damping Factor (15%):** At this level, PageRank values are more evenly distributed across web pages. This is because a lower damping factor decreases dependence on the web pages' link structure, thereby giving more importance to the probability of randomly jumping to a page.
- **High Damping Factor (85%):** A higher damping factor leads to a greater disparity in PageRank values. This scenario favors web pages with more or higher-quality inbound links, thus heavily emphasizing the importance of the network's link structure.

Impact of Iteration Count at High Damping Factor:

- **Iterations with 85% Damping Factor:** The study further investigates how the number of iterations (1, 5, 10, 20) affects the PageRank values, especially at a high damping factor of 85%.
- **Convergence Over Iterations:** With an increase in iterations, PageRank values begin to stabilize. This convergence shows that over time, the influence of initial arbitrary values diminishes, and the ranks increasingly reflect the actual link structure and quality within the web graph.

New Insights and Practical Implications:

- **Dynamic Nature of PageRank:** The findings underscore the flexibility of the PageRank algorithm, where both damping factors and iteration counts can significantly alter the final rankings. This adaptability is beneficial for tailoring the algorithm to specific network characteristics.
- **Algorithm Sensitivity:** The sensitivity of PageRank to its parameters is highlighted, emphasizing the importance of fine-tuning these parameters in different contexts, such as web search.
- **Balance in Parameter Selection:** Selecting the appropriate damping factor and iteration count is crucial. Higher damping factors highlight link structures, but too high a value may disproportionately favor certain pages. More iterations lead to stability but increase computational costs.

In summary, the study reveals that the PageRank algorithm, as applied in Apache Spark, is a potent tool for assessing the relative importance of pages in a network. The algorithm's parameter sensitivity necessitates careful adjustment to achieve optimal performance and meaningful insights in varied contexts.